CSE 102 – Assignment 6

Introduction

This assignment is the first in a series of assignments that will close out the semester on a connected project. What we will be doing is progressively building a health monitoring device (think FitBit or the like, although ours won't look as cool because we aren't going to work so hard on the packaging!).

You may work in pairs of up to two individuals on this assignment.

Pulse Monitoring

The first task is to provide for a capability to monitor heart rate in real time. Pulse oximetry techniques use the light absorptive characteristics of hemoglobin (blood) and the pulsating nature of blood flow in the arteries to determine the oxygen level (or oxygen saturation) in the blood. We won't be going all the way to oxygen saturation levels, but we will use pulse oxygenation probes to measure pulse, or heart rate. Our probe looks like the picture below.

The probe houses a light source (both a red LED and an infrared LED, we will only use the red LED) and a light detector (photodiode). The probe is placed on the finger so there is a light-transmitting path from the source to the detector through a finger. As the blood flow pulsates in the arteries, the amount of light detected varies, and the signal from the light detector is used to measure pulse.

More information about the general technique of pulse oximetry is available here.

Interfacing to the Probe

We have two tasks that must be completed to interface to the probe. First, we must ensure that the red LED is on continuously. Second, we must bias the phototransistor detector (it is really a photodiode, but why complain about manufacturer's documentation? ☺) and read the analog value off the detector. The DB9 connector has the following pin-out.

connector end Pin	Pin Name	Description
9	phototransistor cathode	green wire
5	phototransistor anode	white wire; phototransistor detects level of IR and/or red light transmitted through the finger
7	shield	cable shield, connects to copper shield over the phototransistor
2	LED1	red wire; anode of the IR LED, cathode of the red LED
3	LED2	black wire; cathode of the IR LED, anode of the red LED
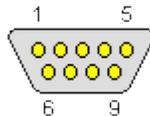
Connect the anode of the red LED (pin 3) to a 100Ω resistor, and connect the other side of the resistor to power (+5 V). Connect the cathode of the red LED ground. This will light the red LED and reverse-bias the infrared LED, so it will stay off.

Next connect the anode of the phototransistor (pin 5) to ground, and connect the cathode of the phototransistor (pin 9) to an analog input pin on the Arduino. In addition, connect a 1MΩ resistor from the phototransistor cathode to power (+5 V). This reverse-biases the detector (which is what is needed to read the light signal) and provides the information to the Arduino analog input.

Reading the Raw Signal

Wrap the sensor around your finger with the LED on the fingernail side. Tape the sides shut so that it is a snug (but not tight) fit. Using appropriate techniques developed earlier in the semester, read the analog input at a sample rate of 500 Hz. Note that you might want to use micros() instead of millis() as your method of accessing the Arduino's free-running timer.

Deliver this streaming data to the PC via the serial port. You will need to decide the format that you use for transmission. Think about what will happen if one byte is lost (say, at startup). Will your serial protocol recover? You might want to increase the speed of the serial port to something faster than the default 9600 baud. (Baud is roughly bit/s, modulo some byte-level framing overhead.)

One option would be to start by using Serial.print() and sending the sample values as ASCII strings. This can be captured on the serial monitor on the PC for further processing (in Java or any other language of your choice).

You need to get to the point where you are reading the stream of raw data in Java so that it can be processed in real time.

Butterworth Bandpass Filter

If you look at a plot of the raw data as a function of time, you will see that it is quite noisy. We must filter the raw data to enable us to discern the heart rate signal that is buried in all that noise. For this assignment, we will use a bandpass filter (it "passes" frequencies within a specified "band") in the family of filter designs originally described by British engineer and physicist Steven Butterworth in 1930.

Computationally, the filter has the following properties. The inputs are a stream of values x_i and the filter outputs are a stream of values y_i where i is continually increasing. Given both current and previous input values as well as previous output values, the current output value is computed with the following difference equation.

$$y_i = b_0 x_i + b_1 x_{i-1} + b_2 x_{i-2} - a_1 y_{i-1} - a_2 y_{i-2}$$

where the values of a_i and b_i are constant coefficients. Author Java code to execute this filter, applied to raw data samples delivered from the Aruduino platform. You may perform this computation entirely using floating point numbers (double in Java).

Appropriate values for a_i and b_i for this filter are listed below:

a(1) = -1.9789, a(2) = 0.97927

b(0) = 0.01035, b(1) = 0, b(2) = -0.01036

Peak Detector

To determine the heart rate, we must figure out what is the period of the filtered raw signals. This can then be converted to a rate. We will measure the period by assessing the time between successive peaks. There are many approaches to peak detection, but the simplest have the basic form

Sample i is a peak if (y(i) > y(i-1)) && (y(i) > y(i+1))

This is frequently augmented by adding a test to ensure that the values are above the mean (i.e., in our case, greater than 0).

Author Java code to detect the peaks in the filtered signal, and convert these peaks into heart rate. Display the result on the PC.

Submitting Your Work

When you have finished your assignment and demonstrated it to the instructor or TA, make sure they record your completion.